

Final Project Report on An Environment for Geometric Object Manipulation and Monitoring

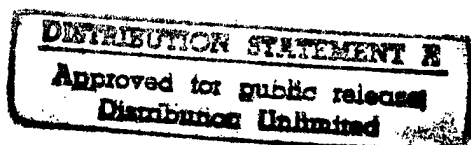
Grant No. N00014-93-1-0272

1997

D. T. Lee
Department of Electrical and Computer Engineering
Northwestern University
Evanston, IL 60208
e-mail: dtlee@ece.nwu.edu
+1-847-491-5007
+1-847-467-4144 (FAX)
URL: <http://www.ece.nwu.edu/~dtlee>

Project Summary

In this project we aim to build a computational framework for visualization of geometric objects as they are manipulated. Specifically we have built an environment that enables the user to design and test geometric algorithms by visualizing the state of an algorithm and monitoring the objects being manipulated. The system is dubbed GEOMAMOS for GEometric Object MANipulation/MOnitoring System. It is an X-window based, menu-driven system and runs under **unix** operating system. A World Wide Web page for this project can be found at <http://www.ece.nwu.edu/~theory/geomamos.html>.



19970722 175

Project Description

Since the inception of this project four years ago, we have successfully built a working prototype that allows the user to manipulate input geometric objects with a mouse, and to visualize the contents of geometric objects in the user's program with a simple GRAPHIC read and GRAPHIC write statements. We have developed an interprocess communication package utilizing socket based UDP/TCP protocol for message passing between the visualization process and the user's program. Geometric object classes and visualization member functions for a limited class of objects have also been developed.

Figure 1 shows an overall block diagram of the system GEOMAMOS. Due to lack of manpower, components GeoDDE, GeoAnimator, and GeoAnalyzer were not implemented. All other modules have been implemented. Let us explain each module briefly.

1. GeoMAMOS Panel - Main control panel.

This is a main user interface of the system. From the *GeoPanel* one can launch one or more GeoSheet processes, and an algorithm browser. It is a centralized control manager responsible for the management of all available GeoSheets and programs in execution. Any GeoSheet or program started from a *GeoPanel* will first be connected to the *GeoPanel*. *GeoPanel* will maintain the information, such as the host machine IP address and the process ID, of the GeoSheet or program. See Figure 2.

2. GeoSheet - Visualization subsystem.

This is a primary component of GeoMAMOS, which is responsible for graphic input and output. It is an interactive visualization tool designed to simplify geometric algorithms visualization procedures in a distributed environment. It is display device independent, although in the current release the display is based on Xfig (Facility for Interactive Generation of Figures under X11)¹ and is primarily for 2-dimensional objects and runs under the Sun Microsystems' environment. It has also been ported to other platforms such as Silicon Graphics IRIS workstations to support 3D graphics, called Geo3DSheet.

The following is a brief summary of the main features of *GeoSheet* that supports geometric algorithm visualization and development.

- *On-Line Visualization Invocation.* This allows visualization operations to be executed directly from user's program. We enhance Xfig with a message driven interface to support such a feature. Compared to other data visualization schemes, this scheme is more tightly coupled with program control flow and provides more flexible visualization support at source code level.
- *Interactive Graphical Object Drawing/Manipulation Capability.* This feature is primarily supported by Xfig that allows the user to draw and manipulate objects interactively in an X window. The user can conveniently use a mouse to create,

¹Smith, B. V., The Xfig User Manual, 1993.

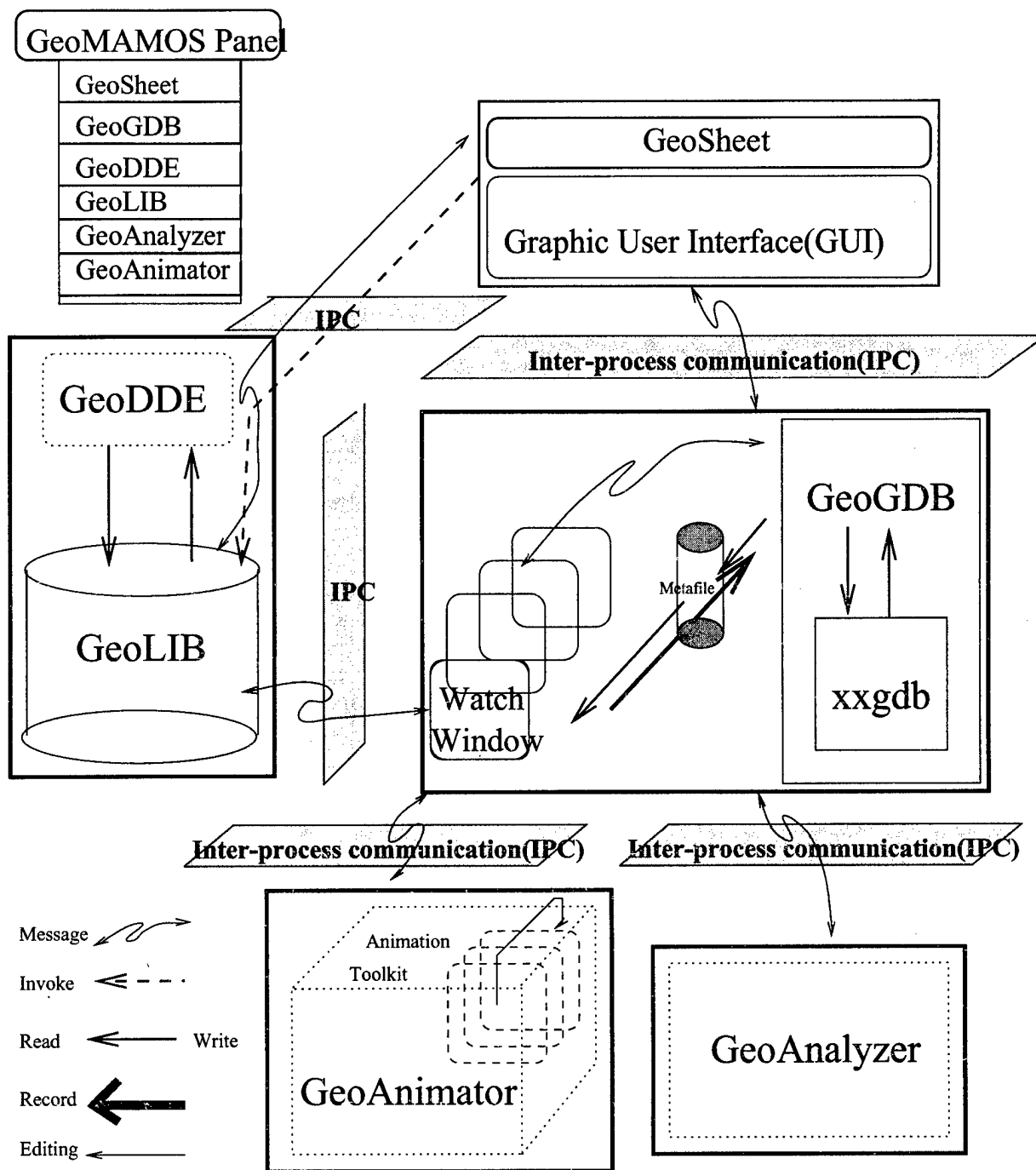


Figure 1: Components of GeoMAMOS

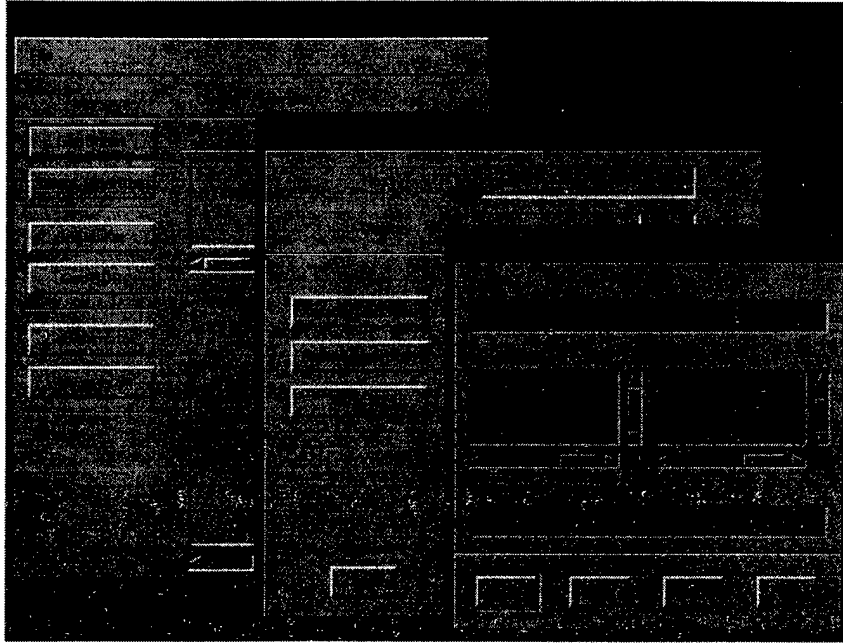


Figure 2: geoPanel window and interaction with other components.

import or edit data objects and export the drawn objects to other GeoSheets, or print out visualized computation results. The ability of importing or exporting visualized objects in Xfig or postscript format is also useful for presenting the computation results.

- *Communication/Comparisons of Algorithms.* GeoSheet is also designed to serve as a communication platform for different algorithms at the program input or output level. For example, an *algorithm pipeline* can be formed by directing the output of one algorithm to be the input of another algorithm. Algorithm pipeline can help the user use available algorithms to explore solutions to new problems. Another usage of GeoSheet is for comparison of different heuristics for NP-complete problems, e.g., Steiner tree problem. We can direct all outputs of the tested heuristics to the same GeoSheet and use different colors or drawing attributes to represent these outputs. Visually we can compare the quality of the output.
- *Distributed GeoSheet Manipulation.* Via Internet service application software can be exchanged through anonymous ftp services. If we know that certain programs are available at remote sites, we can download and execute them locally. However such scheme sometimes requires tedious installation efforts and large storage space. Thus multiple copies of the same software exist. This is not only inefficient in resource usage, but also inconvenient, if we only need to test run a small

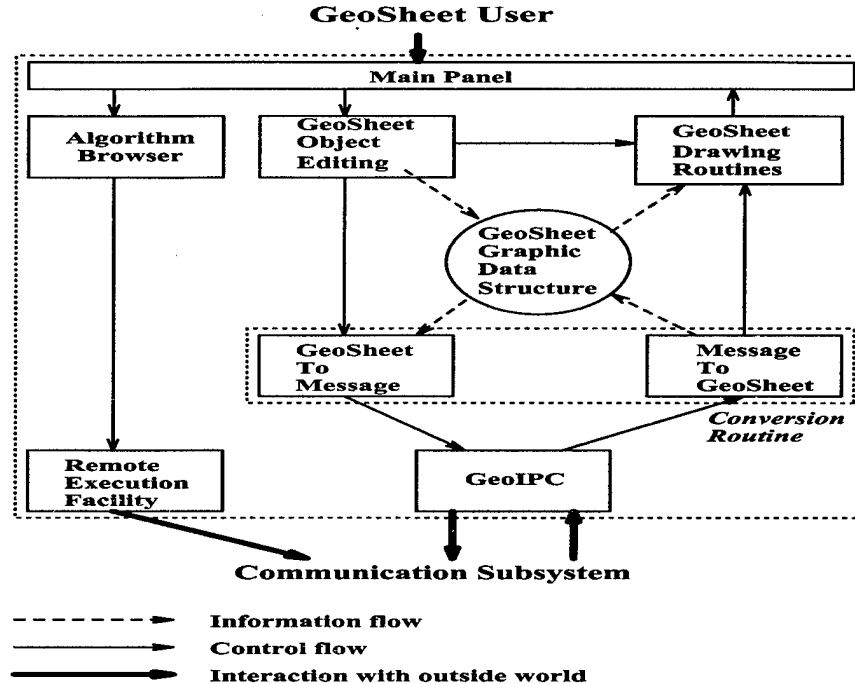


Figure 3: An internal block architecture of GeoSheet.

program. It is worse if the needed programs are available at different sites. This shortcoming can be removed if we allow the user to run those algorithms directly at remote sites and display the results locally. *GeoSheet* is equipped with remote execution facilities to assist the user to execute algorithms at distributed sites. There is a limitation, of course. Since the communication among different sites is done via message passing, reliability of the communication link is crucial. We have successfully tested this facility in our local network but when the amount of data to be transmitted over the Internet, some packets may get lost.

GeoSheet can also be launched on its own without using the *geoPanel*, and with *GeoIPC* it can be executed as a separate process across the network. See Figures 3 and 4 for an illustration of its architecture and Ref. [21] for more details.

3. **GeoGDB** - A front-end X-windows based graphical debugger.

The graphical debugger runs on top of the UNIX source-level debugger, *gdb*. It has a friendly multi-window graphical user interface that serves as the primary communication bridge between the user's program and the underlying debugger. The user can open one or more *watch windows* to observe the contents of different data objects as the program is being executed. The component is to be augmented with a command recorder which will allow the user to plan/edit an animation sequence while

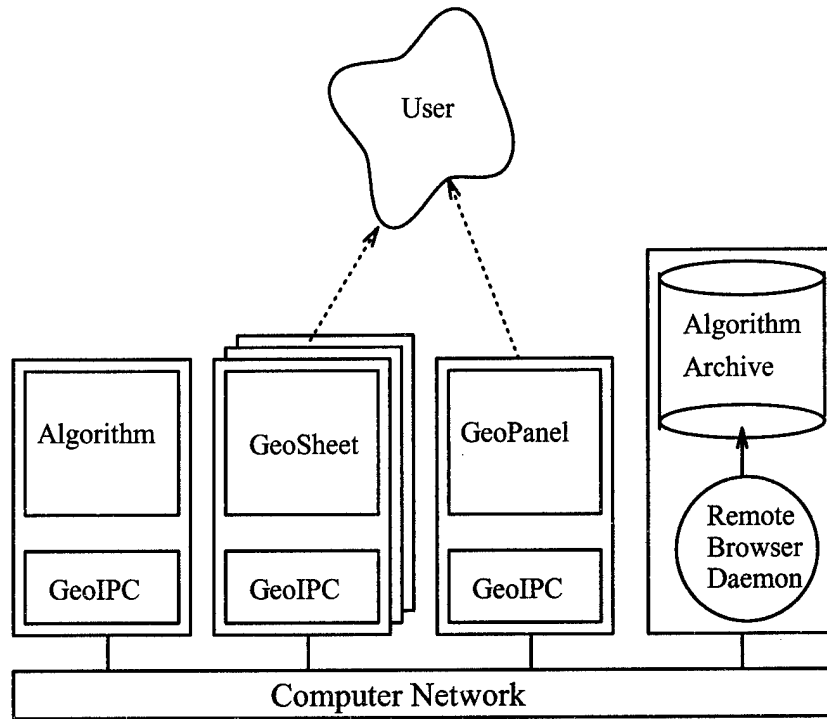


Figure 4: GeoSheet can be run as a separate process in distributed networks.

stepping through the code during the debugging phase. The sequence can be stored in a profile to be re-animated for illustration/demonstration purposes. This is to be interfaced with the component **GeoAnimator**. See Figure 5 for an illustration of the architecture of GeoGDB.

4. **GeoIPC** - Interprocess communication

This component is the backbone of the distributed nature of the system. It is implemented using sockets based on UDP/TCP and serves as a communication vehicle among all processes, including GeoSheet, user's program, GeoPanel, GeoGDB, GeoAnimator and GeoAnalyzer. GeoIPC supports the connection establishment, communication and synchronization among various components. Its functions include connecting distributed GeoSheets and algorithm executions; locating the GeoSheet specified in the **QuerySheet** operation; and providing the message communication utility that supports virtual buffer of logically unlimited length, and the process synchronization and communication for programs and GeoSheet.

5. **GeoLIB** - A library of geometric/graph objects.

Like most of the User Interface objects, the geometric/graph objects of GeoLIB are maintained in a class hierarchy. Such a feature helps the user quickly locate the ap-

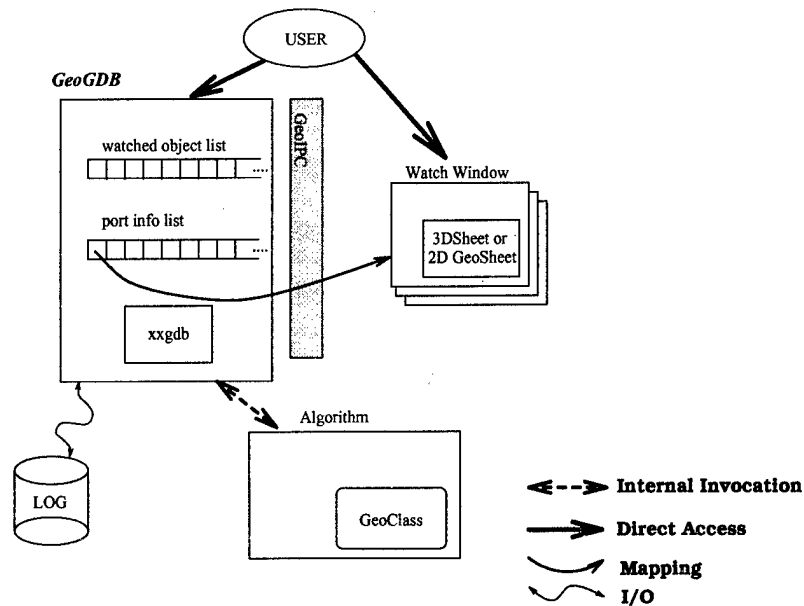


Figure 5: The architecture of GeoGDB

appropriate objects they would like to use in their design. A complex object can be composed of several simple sub-components which are provided by the GeoLIB. GeoLIB is designed to support as many geometric/graph objects as possible; the initial set of supported objects contains only the basic and the most frequently used graph objects. When a new data structure for a particular geometric/graph object is introduced in the algorithm development process, GeoLIB, with the aid of GeoDDE, can be easily extended to support the new object by adding the class definition and related routines.

6. GeoDDE - A data structure diagram editor.

GeoDDE is not yet implemented. It is to provide the user with the ease of building the data structure components of a geometric/graph object. A Class Browser showing the GeoLIB class hierarchy is to be provided with the component. To construct a data structure, one simply draws the structure diagram and specifies each component as either a sub-structure or an object inherited from the GeoLIB class hierarchy. Once the design is complete, a prototype based on the C language (or C++) will be generated. There's no need to implement separate display routines for the new data structure, since they can all be inherited or composed from the built-in classes in GeoLIB.

7. GeoAnimator - An animator builder.

GeoAnimator, which is not yet implemented, is the component for building an animation sequence of a fully debugged program in the GeoMAMOS environment. A **record** button will be supplied in each watch window to activate the recording process of the current display. Each update to the display canvas is saved as an individual frame in

a metafile. And each frame contains only the most recent display information of the object(s) currently shown in the window. A set of frames can be concatenated to form a series of films and later on edited or played back by GeoAnimator.

8. **GeoAnalyzer** - A profiler.

GeoAnalyzer is not yet implemented. It is to provide timing analysis tools for the user's program. The overall running time of the program can be recorded, or certain key operations can be tracked.

In addition to the above routines, a number of geometric algorithms have been implemented with built-in visualization routines and they are all put together under a common subdirectory. This will be used as an initial base for geometric software library. More software packages will be added as soon as they are implemented and tested. Since our geometric software makes heavy use of LEDA^{2 3}, an on-line menu of its functions, which can be found in <http://web.ece.nwu.edu/~theory/leda/> has been prepared to facilitate the implementation of geometric algorithms. The GeoMAMOS environment will help geometric algorithm designers to develop/implement new codes and via WWW further growth and distribution of geometric software can be expected as well.

While most geometric algorithms require a graphical output to show the results, it is a good design practice to separate machine-dependent output modules from the monitoring system. In doing so it can provide a consistent graphic output without requiring the user to develop a user interface to the program. The main execution of the program only needs to be concerned with I/O and internal data manipulation, while the display routines can be executed by the system concurrently. A **salient** feature of the *monitoring* system is the ability to visualize the contents of various geometric objects dynamically. Our system deviates from the traditional ones in that we do not require the usage of metafiles to store the intermediate results and therefore do not rely on a data visualizer to display them. This is done via GeoIPC that sends the contents of the objects directly to the visualization subsystem via messages.

The manipulation/monitoring system supports C as well as C++ on many Unix machines with or without the multi-thread support. Users' programs need not know the data exchanged between the system and each display window, as the monitoring process is set to run independently of the execution of the program. Consequently the program itself is fully portable as C and C++ are portable to many platforms. The monitoring system can also be used to *test* if a certain heuristic or approach to a problem has promise to produce fruitful results or not. Providing a graphics output as the algorithm executes can give us insights into whether a particular heuristic may or may not work. Allowing user interaction during the monitoring process proves useful for heuristic design in improving its performance

²Näher, S., "LEDA - A Library of Efficient Data Types and Algorithms," Max-Planck-institut für informatik, Saarbrücken, Germany, 1992.

³Näher, S. and C. Uhrig, "The LEDA User Manual," Version R 3.2, Tech. Report, MPI-I-95-1-002, Max-Planck-institut für informatik, Saarbrücken, Germany, 1995.

as well. GeoSheet was very helpful in obtaining the result of minimal Steiner tree in the λ -orientation plane[20].

A number of undergraduate and graduate students were involved in the GeoMAMOS project. Several graduate students obtained M.S. degree and one obtained a Ph. D. degree. We have incorporated GeoSheet into our curriculum and introduced undergraduate and graduate students to the notion of geometric computing and visualization.

The following publications, including internal reports, were produced with the support of the project.

1. H. M. Chang, and T. Tabe, "Prototyping of GeoGDB: A Graphical Debugger for Geometric Algorithms," M.S. Project, Dept. of EE/CS, Northwestern University, April 1993.
2. C.-W. Chang and T. C. Shen, "Implementation of GeoLIB for Geometric Manipulation and Monitoring System," M.S. Project, Dept. of EE/CS, Northwestern University, Dec. 1993.
3. D. Z. Chen and D. T. Lee, "All-Pair Shortest Path on Interval/Circular-Arc Graphs," *Proc. 8th Int'l Parallel Proc. Symposium*, Cancun, April 1994, pp. 224-228.
4. D. Fisher, "GeoSheet: A Geometric Input/Output Device," M.S. Project, Dept. of EE/CS, Northwestern University, June 1994.
5. E. Papadopoulou and D. T. Lee, "Shortest Paths in a Simple Polygon in the Presence of "Forbidden" Vertices," *Proc. 6th Canadian Conference on Computational Geometry*, August 1994, 110-115.
6. M. G. Andrews and D. T. Lee, "Parallel Algorithms on Circular-Arc Graphs," *Computational Geometry: Theory and Applications*, 5,3 (1995), 117-141.
7. C. D. Yang, D. T. Lee and C. K. Wong, "Rectilinear Path Problems Among Rectilinear Obstacles Revisited," *SIAM J. Computing*, 24,3, June 1995, 457-472.
8. E. Papadopoulou and D. T. Lee, "Efficient Computation of the Geodesic Voronoi Diagram of Points in a Simple Polygon," *Proc. European Symposium on Algorithms*, September 25-27, 1995, Corfu - Greece, 238-251.
9. M. H. Alsuwaiyel and D. T. Lee, Finding an Approximate Minimum-Link Visibility Path Inside a Simple Polygon," *Info. Processing Letters*, 55 (1995) 75-79.
10. M. J. Atallah, D. Z. Chen and D. T. Lee, "An Optimal Algorithm for Shortest Paths on Weighted Interval and Circular-Arc Graphs with Applications" *Algorithmica*, 14,5, Nov. 1995, 429-441.
11. H.-F. S. Chen and D. T. Lee, "A Faster Algorithm for Rubber-Band Equivalent Transformation for Planar VLSI Layouts," *IEEE Trans. Computer-Aided Design*, 15,2, Feb. 1996, 217-227.

12. D. T. Lee, "Computational Geometry," invited paper, *ACM Computing Surveys*, 28,1, March 1996, 27-31.
13. D. T. Lee, "Geometric Algorithm Visualization, Current Status and Future," *Proc. 1st ACM Workshop on Applied Computational Geometry*, Philadelphia, PA, May 1996, 107 - 112.
14. K. Aoki, "The Prototyping of GeoManager: A Geometric Algorithm Manipulation System," M.S. Project, Dept. of EE/CS, Northwestern University, June 1996.
15. T. S. Hsu, K. H. Tsai, D. W. Wang and D. T. Lee, "Steiner Problems on Directed Acyclic Graphs," *Proc. Second Annual International Computing and Combinatorics Conference*, Hong Kong, June 1996.
16. Y. P. Tseng, "Implementation of a Faster Algorithm for Solving Crossing Minimization Problem," M.S. Project, Dept. of EE/CS, Northwestern University, August 1996.
17. D. T. Lee, "Computational Geometry," in CRC Handbook of Computer Science and Engineering, A. Tucker, Ed., CRC Press 1996, 111-140.
18. B. L. Enke, "Implementation of Minimum Diameter Spanning Tree Algorithms," M.S. Project, Dept. of EE/CS, Northwestern University, Sept. 1996.
19. H. S. Li, "Implementation of Triangulation Refinement Algorithm," M.S. Project, Dept. of EE/CS, Northwestern University, Sept. 1996.
20. D. T. Lee and C. F. Shen, "The Steiner Minimal Tree Problem in the λ -geometry Plane," *Proc. 7th Int'l Symposium on Algorithms and Computation*, Osaka, Japan, Dec. 1996, 247-255.
21. D. T. Lee, S. M. Sheu and C. F. Shen, "GeoSheet: A Distributed Visualization Tool for Geometric Algorithms," *Proc. Int'l Computational Geometry Software Workshop*, Geometry Center, MN, Jan. 18-20, 1995; to appear in *Int'l J. Comput. Geometry & Applications*.
22. K. H. Tsai and D. T. Lee, " k Best Cuts for Circular-Arc Graphs," *Algorithmica*, 18,2 (June 1997), 198-216.

Copyright: GeoSheet: A Distributed Visualization Tool for Geometric Algorithms, Registration number: TXu 695-628, United States Copyright Office, July 24, 1995.